GNUstep

Enhanced architecture

Group 18:
URL: https://youtu.be/oql6jGaAZTo

Group Distribution

- **Lixing Yang**

Group Leader & Abstraction & Introduction and Overview & Use Cases & Lessons Learned

- **Chi Ma**

Presenter & Impacted Files & Plans for testing & PowerPoint & Video & Lessons Learned

- **Tiantian Sang**

Presenter & Current state of the System & PowerPoint & Video & Lessons Learned

- **Nick He**

Conceptual Architecture & SAAM Architecture Analysis & Lessons Learned

- **Dunyi Xie**

Conceptual Architecture & Architecture Analysis

Abstract

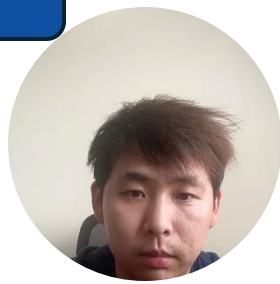
• We propose an AI Code Assistant for Objective-C to enhance GNUstep tools like Gorm.
It offers real-time code suggestions based on UI layout and developer prompts.

• We analyze architectural changes, compare design alternatives using SAAM, and develop a testing plan.
The modular approach is selected for its flexibility, maintainability, and ease of integration.

Introduction and Overview

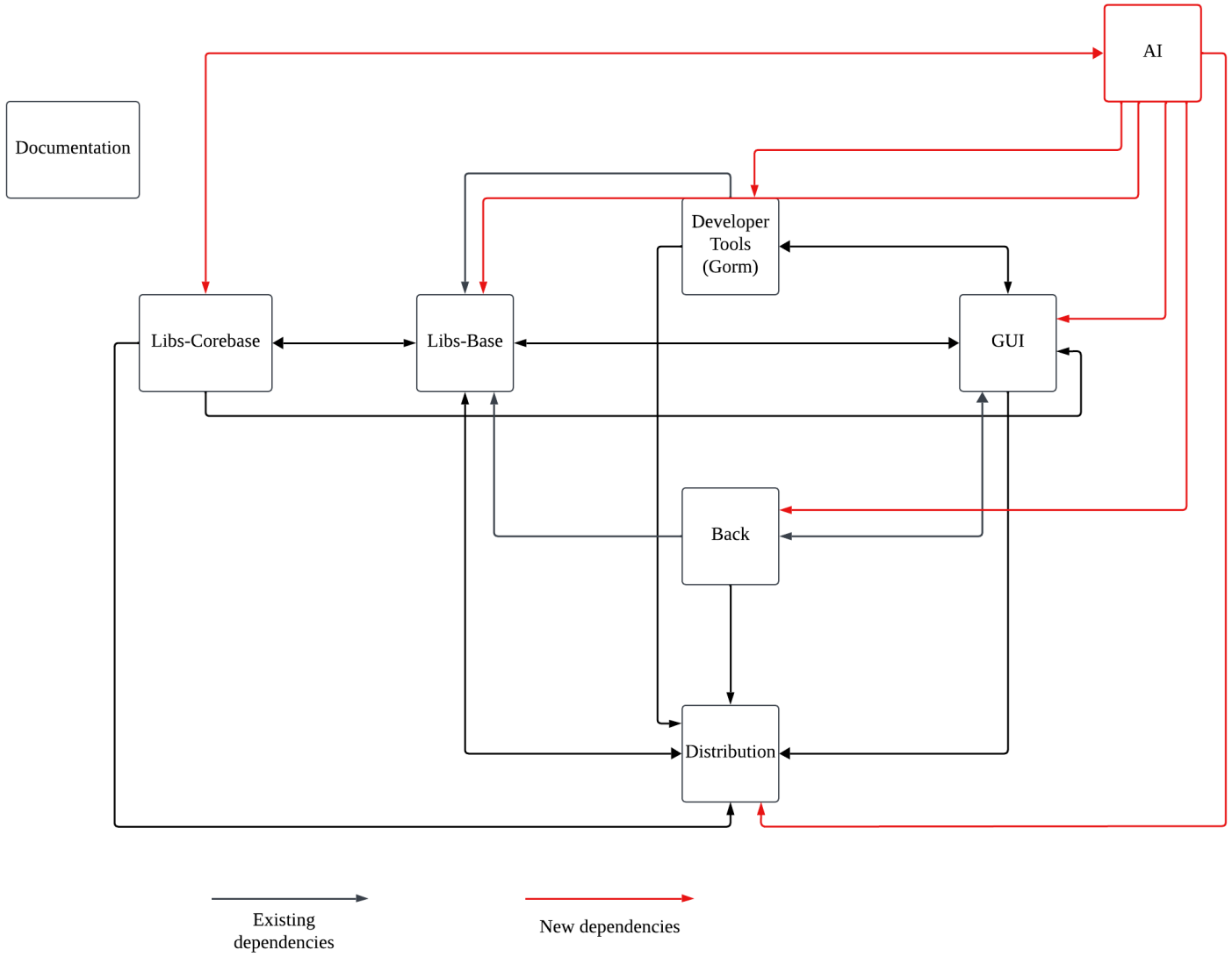
The report is structured as follows:

- Section 1: Abstract
- Section 2: Introduction & Overview
- Section 3: Recap of Conceptual Architecture
- Section 4: The current state of the System
- Section 5: SAAM Architecture Analysis
- Section 6: Impacted Files
- Section 7: Architecture Analysis
- Section 8: Use Case
- Section 3: Planes for testing
- Section 9: Lessons Learned
- Section 9: Reference



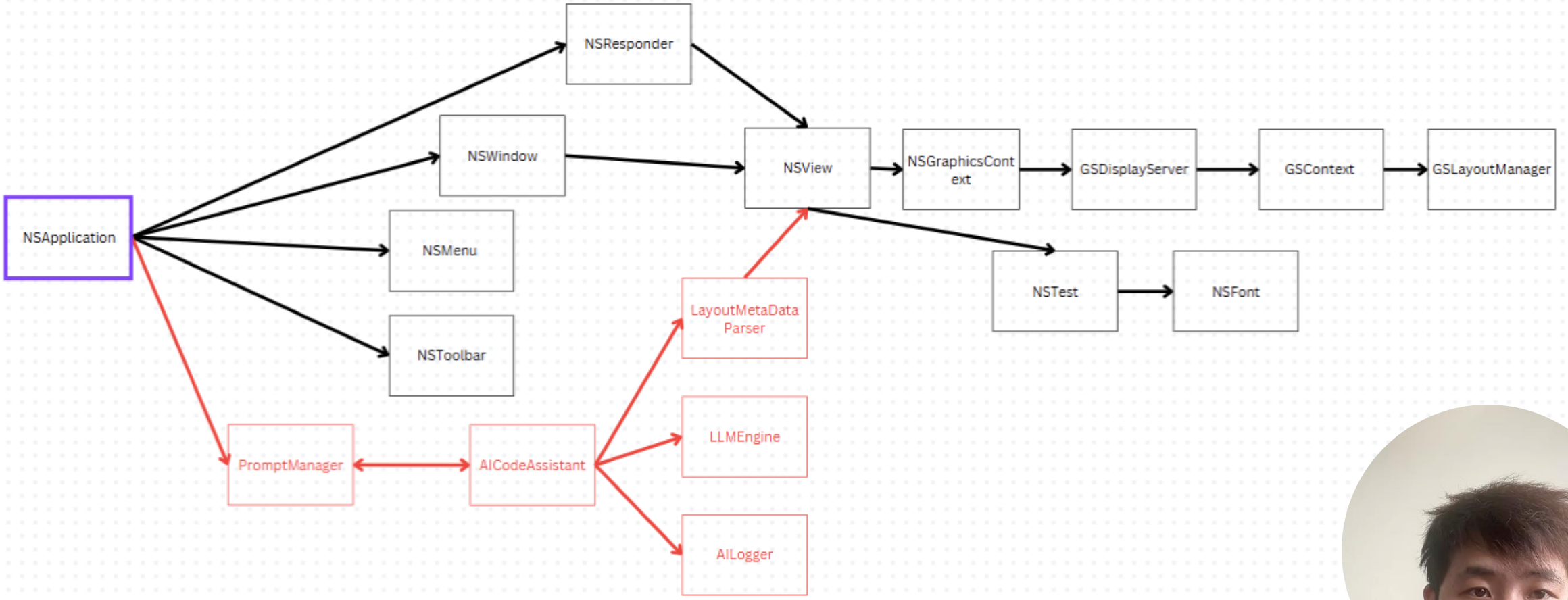
Recap of Conceptual Architecture

High-level conceptual architecture of AI module



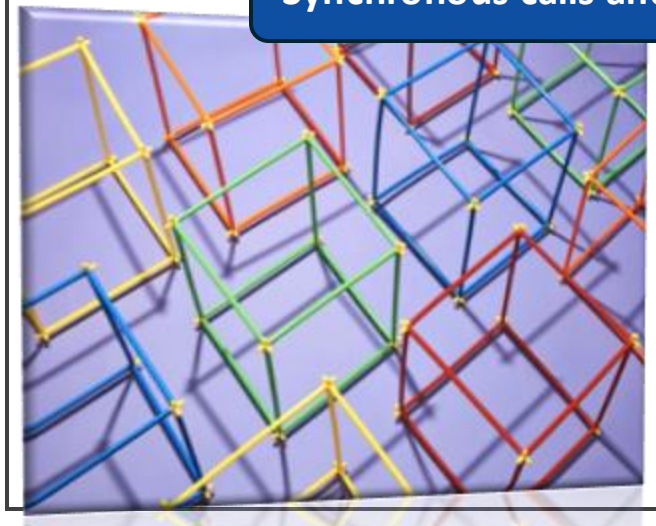
Recap of Conceptual Architecture

Low-level conceptual architecture of AI module



Current state of gnustep

- Manual UI and code management
- Synchronous calls and file manipulation



- No Metadata Extraction
- No Prompt Management
- No JSON/XML

Visison for enhancement

Increased effort and time

Reduced innovation

Lower system responsiveness

Evolution and maintainance

Modular, asynchronous

Standardized protocols

Real-time metadata extraction and code generation.

Improved flexibility, reliability, and future scalability.

SAAM Stakeholders & Alternatives

Stakeholders & Key Concerns

GNUstep Developers:

want low integration overhead, robustness, testability

GNUstep Users:

expect fast, accurate, and optional AI suggestions

Project Managers:

value stability, on-time delivery, and modular design

AI Integrators:

need flexibility, backend independence, easy upgrades

Two Design Approaches

Embedded AI Assistant

Directly integrated into Gorm / IDE

✅ Pro: Simple to implement

❌ Con: Tight coupling, low maintainability

Modular AI Subsystem

Separate module communicating via API

✅ Pro: Easy to test, reusable, scalable

❌ Con: Slightly more design effort

SAAM Stakeholders & Alternatives

SAAM Evaluation Table:

NFR	Embedded	Modular
Maintainability	✗ Negative	✓ Positive
Evolvability	○ Neutral	✓ Positive
Testability	○ Neutral	✓ Positive
Performance	✓ Positive	○ Neutral
Reusability	✗ Negative	✓ Positive
Stability	✗ Negative	✓ Positive

Impacted Files for AI Assistant Integration

Gorm (UI Builder)

- GormController.m/.h: Collect user input & insert AI-generated code
- GormDocument.m/.h: Display and manage code suggestions (with undo/redo)
- GormPreferences.m/.h: Add AI assistant settings (e.g., enable/disable)

GUI Library (libs-gui)

- NSView.m/.h: Extract UI element metadata to improve code suggestions

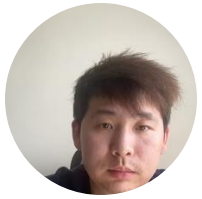
Core Logic

- CodeAssistantRequest.* (new): Prepare and send prompts to the assistant
- CodeAssistantResponse.* (new): Parse and validate AI responses

Core Logic

- ProjectFileEditor.m/.h: Enable AI suggestions during code editing
- ProjectCenterMenu.m/.h: Add menu actions like “Ask AI” or “Generate Method”

Architecture Analysis of AI Code Assistant



Client-Server Architecture
AI Assistant runs as a separate process or on a remote server

AI Assistant runs as a separate process or on a remote server

GNUstep app (Gorm/IDE) acts as client, sends prompt requests via API

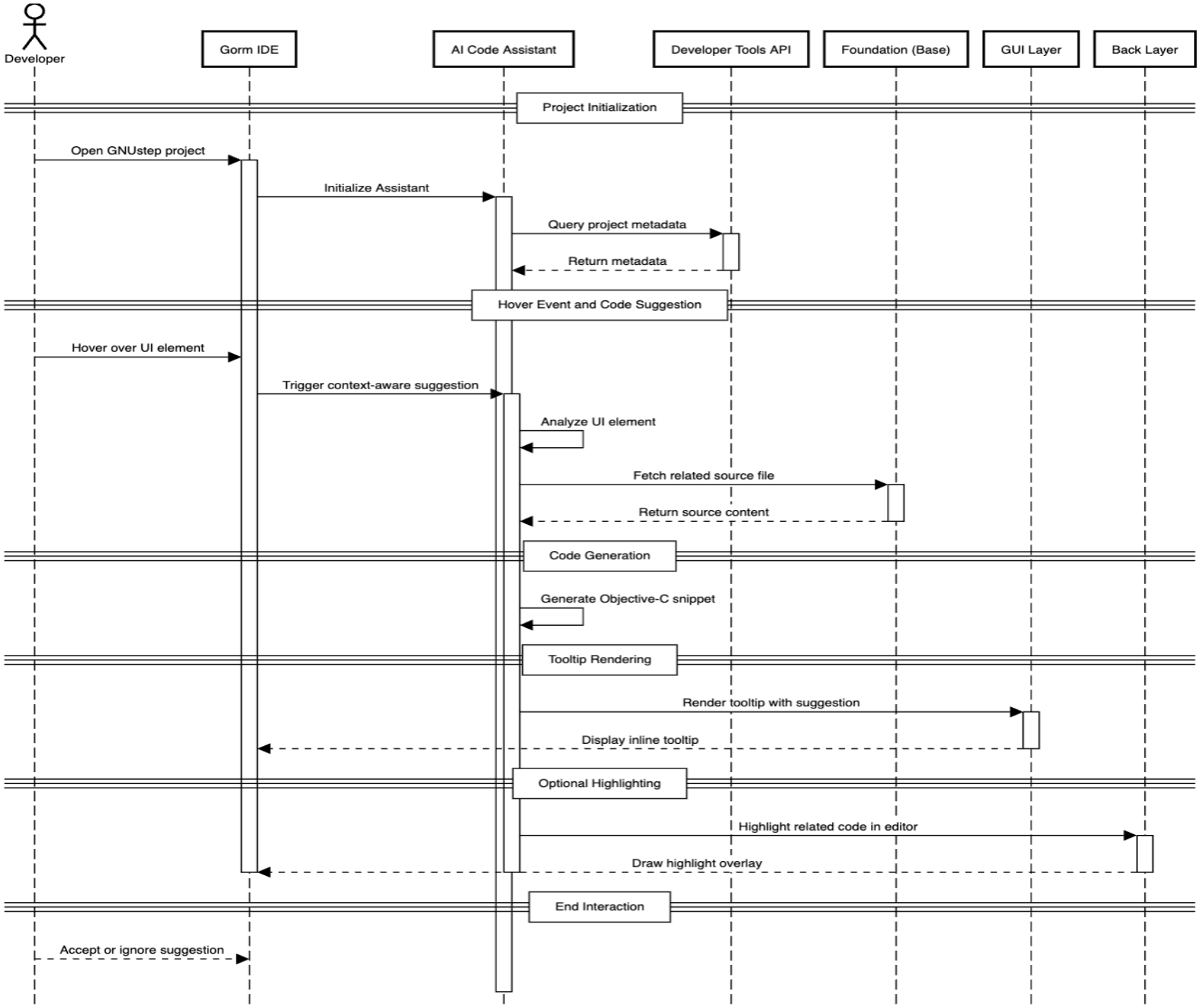
Scalable and flexible: AI backend can be updated without changing GUI

Client-Server Architecture

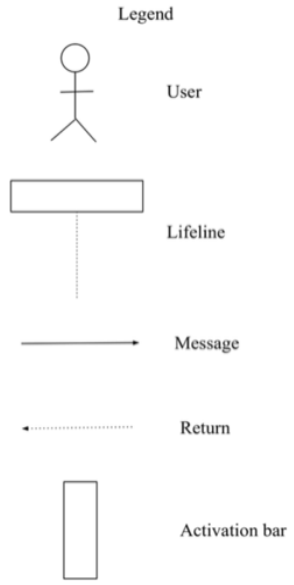
Client-Server Architecture

Client-Server Architecture

Layered Architecture
All functionality exists in **one app**, split into



Use case: context-aware code suggestion



Testing Plan for AI Assistant Integration

Unit Testing

- Test prompt building & code parsing individually
- Validate AI outputs & handle syntax errors

Integration Testing

- Gorm ←→ Assistant: check info flow and code insertion
- ProjectCenter ←→ File editor: ensure proper placement

User Interaction Testing

- Evaluate suggestion accuracy and readability
- Confirm preference toggles and user controls work as expected

Performance Testing

- Measure response time
- Test behavior on large-scale projects

Error Handling

- Handle missing inputs, empty responses, and offline AI service
- Avoid system crashes and show user-friendly errors

Security & Stability

- No data leaks without consent
- Prevent Gorm/IDE crashes from assistant failures

lesson Learned